정렬 알고리즘 보고서

1815060 문정현

1. 코드

```
#include "pch.h"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define NUMOFNUMBERS 500000
// 5만개, 7.5만개, 10만개, 12.5만개, ..., 50만개까지 증가시키며 시간 측정할 것

long randNumbers[NUMOFNUMBERS];
 //무작위 숫자들을 저장하는 공간
long temp[NUMOFNUMBERS];
//병합 정렬과 계수 정렬에서 정렬된 것을 저장하기 위한 공간
long C[RAND_MAX];
//계수 정렬을 위한 횟수 세는 공간

// 선택 정렬 함수
void selectionSort(long A[], long n);
long theLargest(long A[], long last);

// 병합 정렬 함수
void mergeSort(long A[], long p,long r);
//정렬할 배열 A
//배열의 첫번째 인덱스 p
//마지막 인덱스 r
void merge(long A[], long p,long q,long r);
//배열의 중간 인덱스 q

//계수 정렬 함수
void countingSort(long A[],long B[],long n);
//정렬할 배열 A와 배열을 위한 여분의 배열 B
//정렬할 배열의 원소 갯수 n
```

- 선택 정렬

```
void selectionSort(long A[], long n) {
	int last, k, tmp;
	for (last = n - 1; last >= 1; last--) {
		k = theLargest(A, last);
		// A[0...last] 중 가장 큰 수 A[k] 찾기
		tmp = A[last];
		A[last] = A[k];
		A[k] = tmp;
```

```
        }
}
long theLargest(long A[], long last) {
        long largest, i;
        largest = 0;
        for (i = 1; i <= last; i++){
            if (A[i] > A[largest]){ largest = i; }
        }
        return largest;
}
```

- 병합 정렬

```
void merge(long A[], long p, long q, long r){
        long i=p;//좌측 배열의 인덱스
        long j=q+1;//우측 배열의 인덱스
        long k=p;//정렬 배열의 인덱스

        while((i<=q)&&(j<=r)){
              if(A[i]<A[j]){temp[k++] = A[i++];}
              else{temp[k++] = A[j++];}
         //두 배열들을 비교하면 작은 원소를 정렬 배열에 넣고
 //해당 원소가 있던 배열의 인덱스를 증가시킨다.
        }

        //하나의 배열만 남은 경우 정렬 배열에 모두 넣는다
        if(i>q){while(j<=r){temp[k++] = A[j++];}}
        if(j>r){while(i<=q){temp[k++] = A[i++];}}

        //정렬된 배열을 복사
         for(k=p;k<r+1;k++){ A[k]=temp[k]; }
}
void mergeSort(long A[], long p,long r){
        long q; //배열을 나누기 위한 중간 인덱스
        if(p<r){
              q = abs((p+r)/2);
              mergeSort(A,p,q);
              //좌측 배열에 대한 병합정렬을 재귀적으로 실행
             mergeSort(A,q+1,r);
             //우측 배열에 대한 병합 정렬을 재귀적으로 실행
             merge(A,p,q,r);//두 정렬된 좌우 배열에 대한 병합
        }
}
```

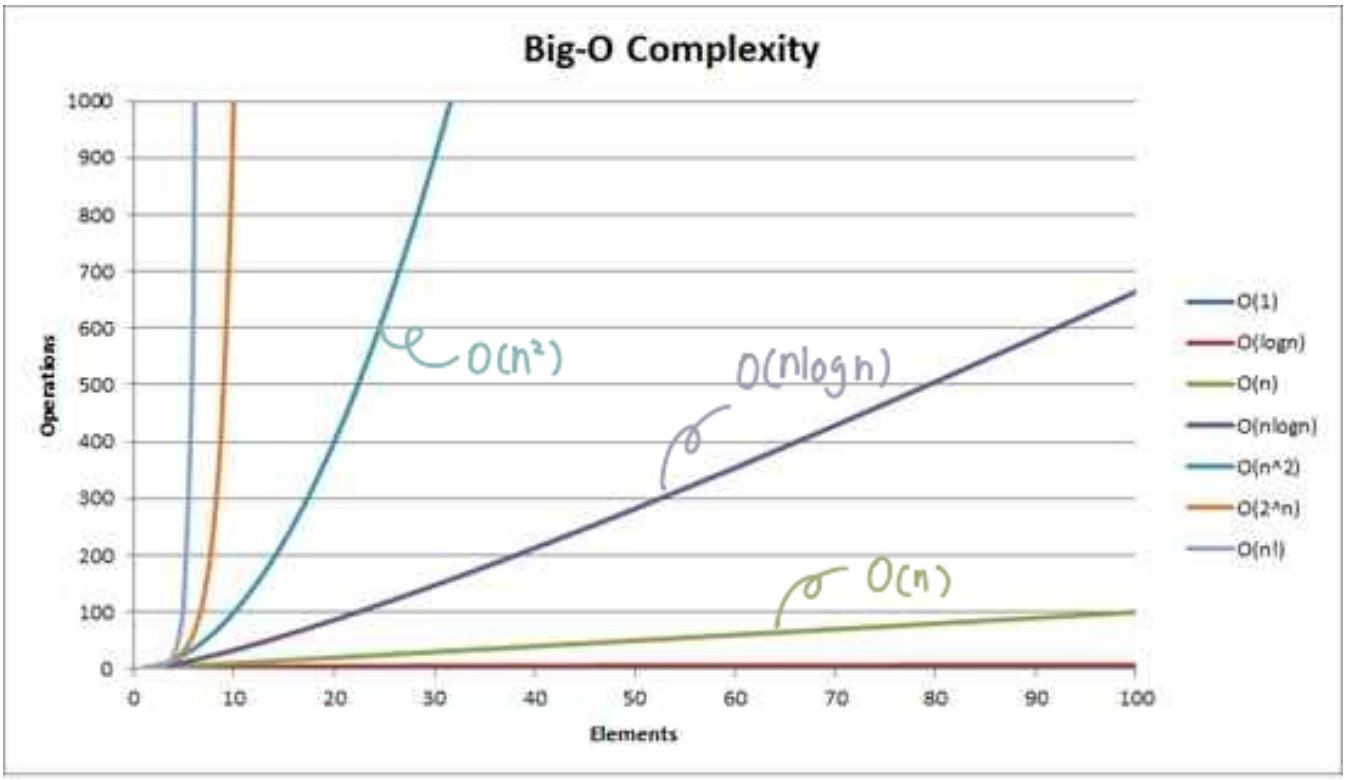
- 계수 정렬

```
void countingSort(long A[],long B[],long n){
        int i;
```

```
        long tmp1,tmp2;

        for(i=0;i<n;i++){
              tmp1 = A[i];
              ++C[tmp1];
        }
       // A[i]의 값이 나오는 횟수를 C[A[i]]에 저장한다

  for(i=1;i<RAND_MAX;i++){ C[i] += C[i-1]; }
     //C[i]에는 i보다 작거나 같은 것들의 횟수가 저장된다.

     for(i=0;i<n;i++){
           tmp1 = A[i];
           tmp2 = C[tmp1];
           B[--tmp2] = A[i];
           --C[tmp1];
           // C에 저장된 정보를 이용하여
           //해당 인덱스에 값을 넣어 정렬시킬 배열에 넣는다
        }
        //정렬된 배열 복사
        for(i=0;i<n;i++){A[i] = B[i];}
}
```

2. 복잡도 그래프(이론 vs 실제 실행)

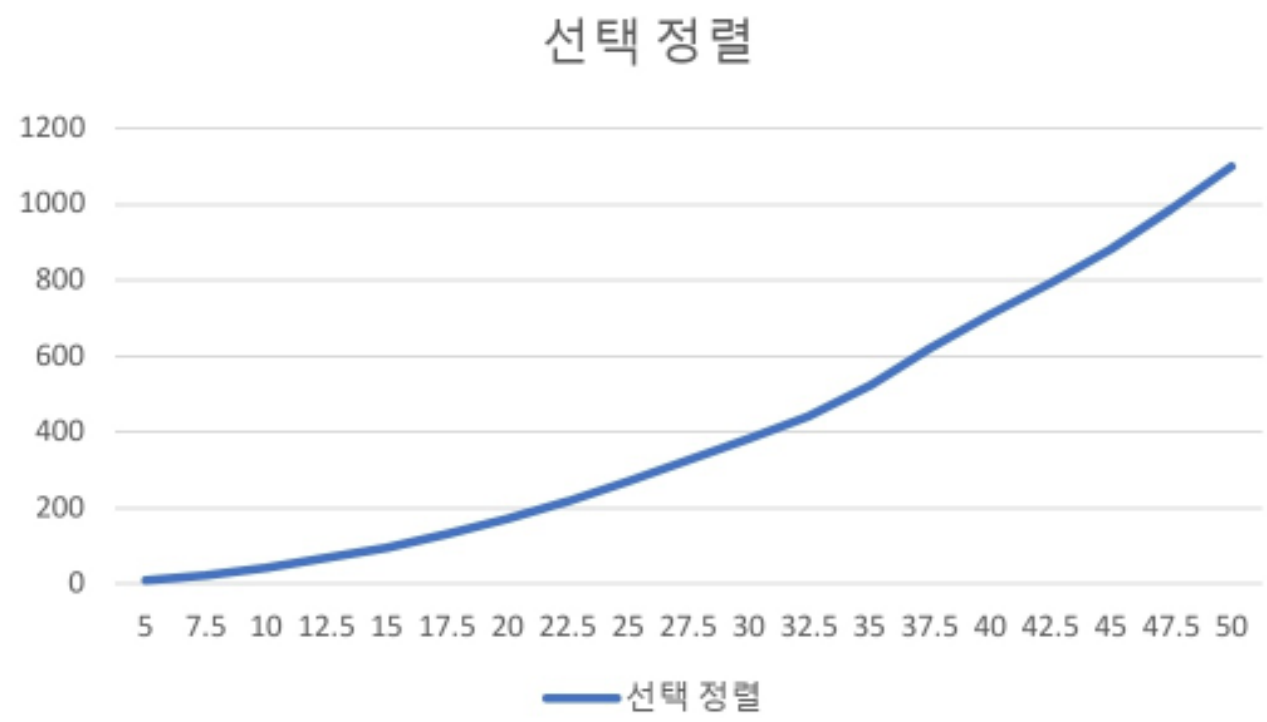
- 이론

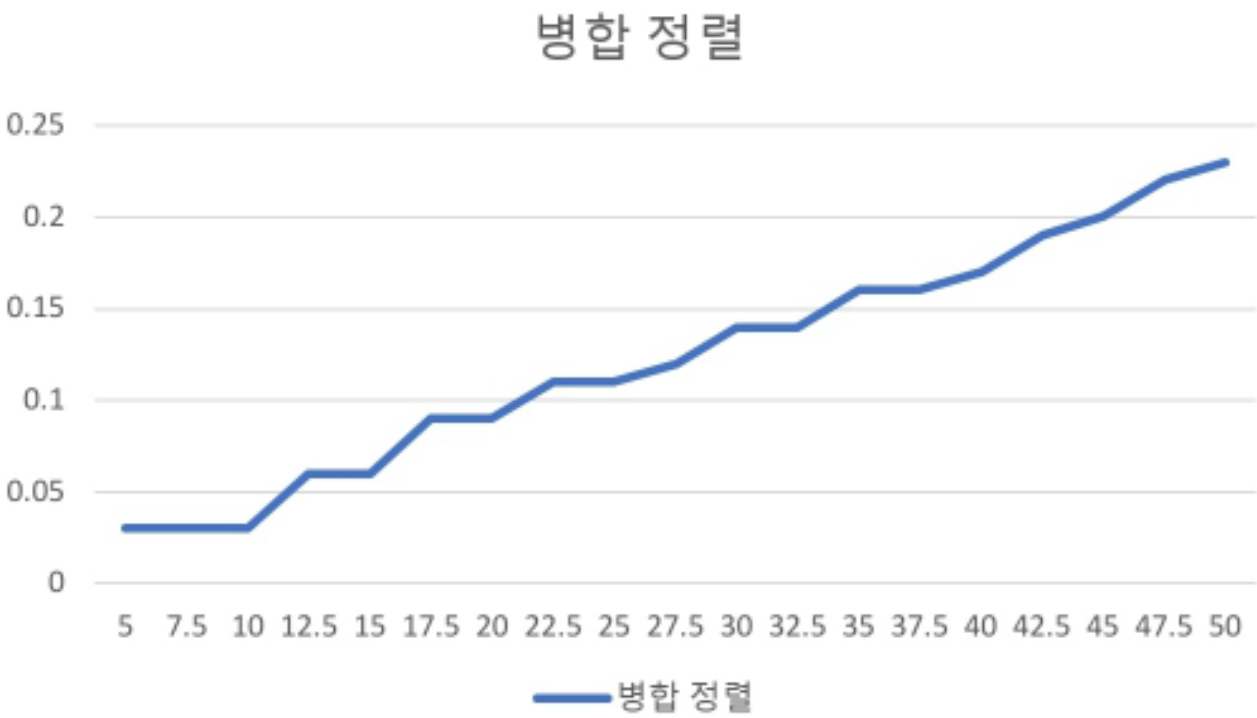


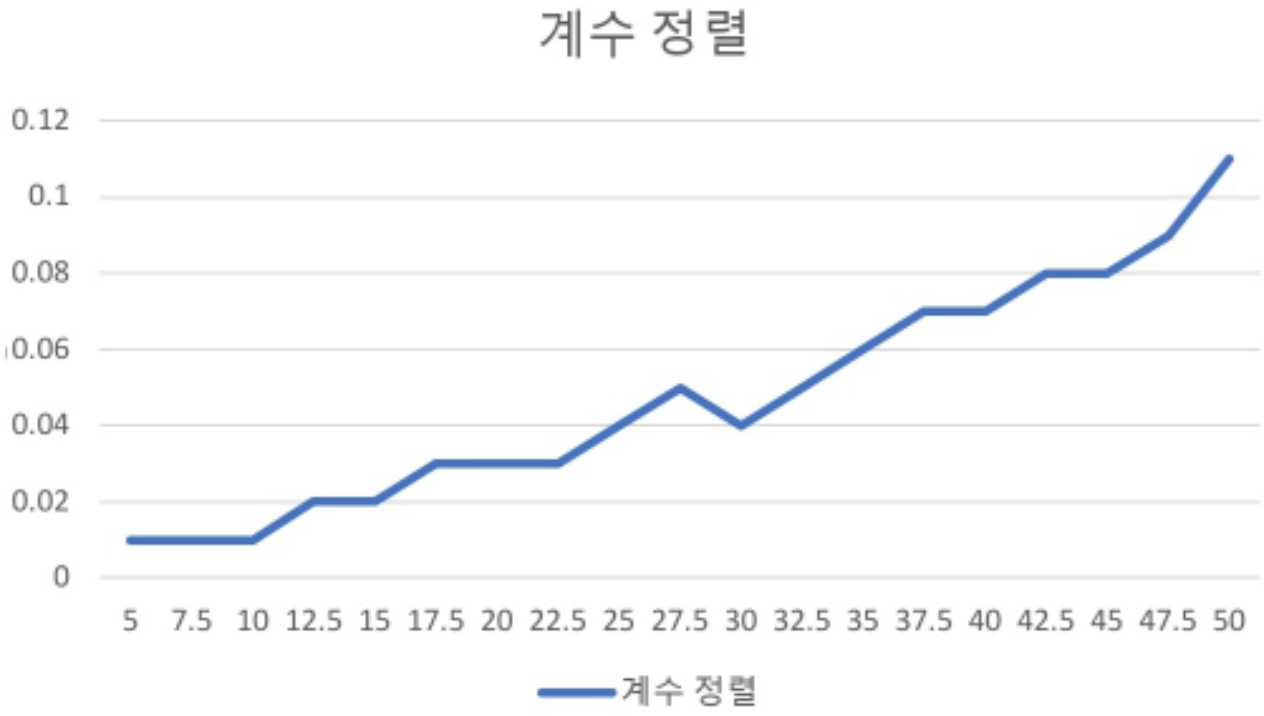
1. 선택 정렬: O(n^2)

2. 병합 정렬: O(nlogn)
3. 계수 정렬: O(n)

- 실제 실행






계수 정렬
0.12
0.1
0.08
0.06
0.04
0.02
0
5 7.5 10 12.5 15 17.5 20 22.5 25 27.5 30 32.5 35 37.5 40 42.5 45 47.5 50
계수 정렬